Ex. BINARY SEARCH

LET A BE AN ARRAY OF INTEGERS (SAY) WITH $n = \text{length}[A]$. WE WILL ADOPT THE CONVENTION THAT ARRAY INDICES BEGIN AT 1. THUS

$$A[1 \cdots n] = (A[1], \ldots, A[n]).$$

LET $A[p \cdots r]$ DENOTE THE SUBARRAY

$$A[p \cdots r] = (A[p], \ldots, A[r]).$$

IF $p > r$ WE UNDERSTAND THIS TO BE AN EMPTY ARRAY.

ASSUME $A[1 \cdots n]$ IS SORTED IN INCREASING ORDER (WITH POSSIBLE REPEATED ELEMENTS.)

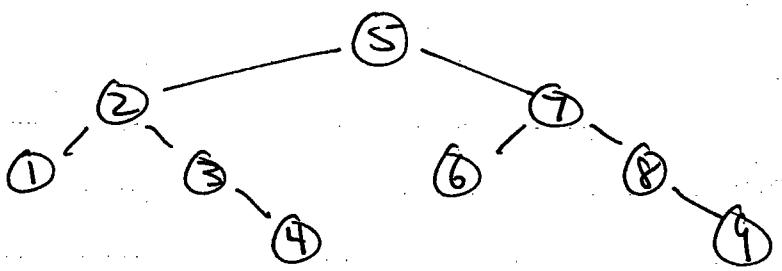
BINARY SEARCH IS A D&C ALGORITHM WHICH LOCATES A GIVEN TARGET t IN THE SUB-ARRAY $A[p \cdots r]$. IF AN INDEX i IS FOUND SUCH THAT $A[i] = t$, THEN i IS RETURNED, OTHERWISE 0 IS RETURNED.

BinSearch (A, p, r, t) (PRE: $A[p..r]$ SORTED.)

1.) if $p > r$
2.) return 0
3.) $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
4.) if $A[q] = t$
5.) return q
6.) if $A[q] < t$
7.) return BinSearch$(A, q+1, r, t)$
8.) return BinSearch$(A, p, q-1, t)$

Ex. $A = (1, 2, 3, 4, 5, 6, 7, 8, 9)$

1 2 3 4 5 6 7 8 9



THIS BINARY SEARCH TREE REPRESENTS THE ORDER IN WHICH THE TARGET t IS COMPARED TO ELEMENTS OF A.

THE CALL TO BinSearch ON THE FULL ARRAY $A[1..n]$ IS JUST

BinSearch $(A, 1, n, t)$

THEOREM (CORRECTNESS OF BinSearch)
BinSearch RETURNS EITHER AN INDEX j SUCH THAT $A[j] = t$, OR RETURNS 0 IF NO SUCH j EXISTS.

PROOF
USE INDUCTION ON $m = r - p + 1 = length[A[p \cdots r]]$.

I. BASE
If $m = 0$ THE SUBARRAY DOES NOT CONTAIN TARGET t. ALSO $m = 0$ IMPLIES $r = p - 1 < p$, SO THAT 0 IS RETURNED ON LINE 2.

II. (STRONG) INDUCTION
LET $m > 0$ AND ASSUME THAT BinSearch RETURNS THE CORRECT INDEX ON ANY SUBARRAY OF LENGTH LESS THAN m.

NOW $m > 0$ IMPLIES $r > p - 1$ WHENCE $r \geq p$. THUS $q = \lfloor \frac{p+r}{2} \rfloor$ (LINE 3) IMPLIES $p \leq q \leq r$.

IF $A[q] = t$ THEN OBVIOUSLY BinSearch RETURNS CORRECT INDEX ON LINE 5.

IF $A[q] < t$ THEN $A[(q+1) \cdots r]$ IS A SUBARRAY OF length

$$r - (q+1) - 1 = r - q \leq r - p < r - p + 1 = m.$$

THE INDUCTION HYPOTHESIS GUARANTEES THAT A CORRECT VALUE IS RETURNED ON LINE 7.

IF ON THE OTHER HAND, $A[q] > t$ THEN $A[p \ldots (q-1)]$ IS OF LENGTH

$$(q-1)-p+1 = q-p \leq r-p < r-p+1 = m,$$

SO THE INDUCTION HYPOTHESIS INSURES THAT A CORRECT VALUE IS RETURNED ON LINE 8.

IN ALL CASES A CORRECT VALUE IS RETURNED, AND THE PROOF IS COMPLETE. ///

THE (WORST CASE) ANALYSIS OF BINARY SEARCH IS VERY SIMPLE

$$T(n) = \begin{cases} \Theta(1) & n = 0 \\ 1 \cdot T\left(\frac{n}{2}\right) + \Theta(1) & n \geq 1 \end{cases}$$

$n^{\log_2 1} = n^0 = 1 = \Theta(1)$, SO CASE 2 OF THE MASTER THEOREM SAYS

$$T(n) = \Theta(\lg n).$$

EX MERGE SORT

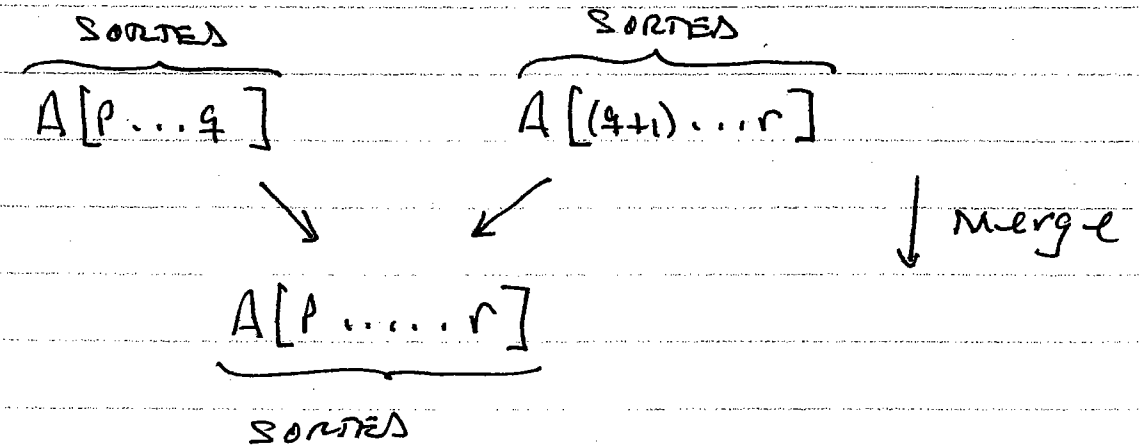
AS BEFORE A is AN ARRAY (OF NUMBERS) WITH $n = length[A]$.

MergeSort RECURSIVELY SORTS A SUBARRAY $A[p..r]$ OF $A[1...n]$ WHERE $1 \le p \le r \le n$.

MergeSort(A, p, r)

1.) if $p < r$
2.) $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3.) MergeSort(A, p, q)
4.) MergeSort(A, q+1, r)
5.) Merge(A, p, q, r)

Merge(A, p, q, r) is A SUB-ALGORITHM WHICH COMBINES THE SORTED SUB ARRAYS $A[p..q]$ AND $A[(q+1)..r]$ INTO ONE SORTED SUBARRAY $A[p..r]$.



TO SORT THE ENTIRE ARRAY $A[1 \cdots n]$ WE CALL MergeSort$(A, 1, n)$.

THEOREM.
AFTER MergeSort(A, p, r) IS CALLED, THE SUB-ARRAY $A[p..r]$ IS SORTED.

PROOF.
WE USE INDUCTION ON $m = r - p + 1 = length[A[p..r]]$.

I. IF $m = 1$ THEN $r = p$ AND $A[p..r]$ IS ALREADY SORTED. INDEED, THE ALGORITHM DOES NOTHING IN THIS CASE.

II. LET $m > 1$ AND ASSUME THAT ANY CALL TO MergeSort ON A SUBARRAY OF LENGTH LESS THAN m RESULTS IN THAT SUBARRAY BEING SORTED.

NOW $m > 1$ IMPLIES $r > p$, SO LINE 2 IS EXECUTED SETTING $q = \lfloor \frac{r+p}{2} \rfloor$. THIS IMPLIES $p \le q < r$:

$$p < r \Rightarrow \begin{cases} 2p < p + r \Rightarrow p < \frac{p+r}{2} \Rightarrow p \le \lfloor \frac{p+r}{2} \rfloor \\ p + r < 2r \Rightarrow \frac{p+r}{2} < r \Rightarrow \lfloor \frac{p+r}{2} \rfloor < r \end{cases} \Rightarrow p \le q < r$$

Thus

$$\text{length}[A[p..q]] = q-p+1 < r-p+1 = m$$

And

$$\text{length}[A[(q+1)..r]] = r-(q+1)+1 = r-q \le r-p < r-p+1 = m$$

The induction hypothesis therefore insures that $A[p..q]$ and $A[(q+1)..r]$ are sorted after lines 3 and 4 are executed. Hence $A[p..r]$ is sorted after the execution of line 5.

This completes the proof. ///

Analysis (worst case)

$$T(n) = \begin{cases} \Theta(1) \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + \Theta(n) \end{cases}$$

This can be simplified to $T(n) = 2T(\frac{n}{2}) + \Theta(n)$, and the Master Theorem (case 2) gives

$$T(n) = \Theta(n \lg n).$$

Ex. Quicksort. (7.1 - 7.4)

AGAIN $A[p \ldots r]$ IS SORTED RECURSIVELY.

Quicksort (A, p, r)

1.) if $p < r$
2.) $\quad q \leftarrow$ Partition (A, p, r)
3.) $\quad$ Quicksort $(A, p, q-1)$
4.) $\quad$ Quicksort $(A, q+1, r)$

Partition (A, p, r)

1.) $i \leftarrow p-1$
2.) for $j \leftarrow p$ to $(r-1)$
3.) $\quad$ if $A[j] \leq A[r]$
4.) $\quad\quad i \leftarrow i+1$
5.) $\quad\quad A[i] \leftrightarrow A[j]$
6.) $A[i+1] \leftrightarrow A[r]$
7.) return $(i+1)$

THE SUBROUTINE Partition(A, p, r) RE-ARRANGES $A[p \ldots r]$ INTO TWO (POSSIBLY EMPTY) SUBARRAYS $A[p \ldots (q-1)]$ AND $A[(q+1) \ldots r]$ SUCH THAT

$$\underbrace{A[p \ldots (q-1)]}_{\text{NOT SORTED}} \leq A[q] \leq \underbrace{A[(q+1) \ldots r]}_{\text{NOT SORTED}}$$

$A[q]$: PIVOT

Ex. Partition

i j=p r-1 r

8 6 1 3 7 2 5 (4) ← pivot

i j

8 6 1 3 7 2 5 (4)

i j

8 6 1 3 7 2 5 (4)

i j

1 6 8 3 7 2 5 (4)

i j

1 6 8 3 7 2 5 (4)

i j

1 3 8 6 7 2 5 (4)

i j

1 3 8 6 7 2 5 (4)

i j

1 3 8 6 7 2 5 (4)

i j

1 3 2 6 7 8 5 (4)

i j

1 3 2 6 7 8 5 (4)

i j

1 3 2 6 7 8 5 (4)

(1 3 2) (4) (7 8 5 6)

A[p..(q-1)] ↑ pivot A[(q+1)..r]

REMARKS.

LOOP 2-5 MAINTAINS THE FOLLOWING INVARIANTS

(i) $i \le j < r$

(ii) $A[p..i] \le A[r]$

(iii) $A[r] \le A[(i+1)..(j-1)]$

(iv) THE ELEMENTS IN $A[i..(r-1)]$ HAVE NOT BEEN PROCESSED AND MAY BE $\ge A[r]$ OR $\le A[r]$.

EXERCISE

- VERIFY THESE CLAIMS ON EXAMPLES
- PROVE THEM.

IT FOLLOWS THAT WHEN PARTITION RETURNS:

$$A[p..(q-1)] \le A[q] \le A[(q+1)...n]$$

THE RUN TIME OF PARTITION (IN TERMS OF WORST AND AVERAGE CASES) IS $\Theta(m)$ WHERE $m = r-p+1$, SINCE LOOP 2-5 STEPS THROUGH THE ENTIRE SUBARRAY $A[p..(r-1)]$, THEN THE PIVOT IS SET IN PLACE.

EXERCISE.
PROVE THE CORRECTNESS OF QUICKSORT BY INDUCTION ON THE LENGTH OF THE SUB-ARRAY $A[p..r]$: $m = r-p+1$.

THE RUNTIME OF QUICKSORT DEPENDS HEAVILY ON THE VALUE q RETURNED BY PARTITION. IF THE SUBARRAYS $A[p..(q-1)]$ AND $A[(q+1)..r]$ ARE NOT BALANCED (i.e. OF ROUGHLY EQUAL SIZE) THEN PERFORMANCE IS INHIBITED. IN THIS CASE ONE RECURSIVE CALL TO QUICKSORT IS ON A SUBARRAY WHICH IS INORDINATELY LONG.

THE WORST CASE OCCURS WHEN THE ARRAY IS ALREADY SORTED. THEN PARTITIONS RETURNS

$$\underbrace{A[p \ldots (r-1)]}_{\text{SORTED}} \leq A[r] \leq \ldots \text{EMPTY} \ldots$$
$$\uparrow\ q$$

LET $T(n)$ DENOTE THE WORST CASE RUN TIME OF QUICKSORT (i.e. WITH $A[1..n]$ ALREADY SORTED.)